International Journal of Control
Vol. 00, No. 00, Month 200x, 1–28

RESEARCH ARTICLE

A PID tuning approach to find the optimal compromise among robustness, performance and control effort. Implementation in a free software tool

Roberto Sanchis* and Ignacio Peñarrocha

Departament d'Enginyeria de Sistemes Industrials i Disseny, Universitat Jaume I, Castelló, Spain.
(*Received 00 Month 200x; final version received 00 Month 200x*)

*Corresponding author. Email: rsanchis@uji.es

ISSN: 0020-7179 print/ISSN 1366-5820 online

DOI: 10.1080/0020717YYxxxxxxxx
http://www.informaworld.com

In this paper we present a PID tuning approach for achieving an optimal compromise between robustness, performance and control effort in terms of measurement noise amplification. The tuning strategy is based on the concept of fixed robustness tuning line, in which the derivative filter parameter is a key point in finding the compromise between control effort due to noise and performance. A model of the plant is assumed to be known, in the form of a transfer function. The approach allows us to solve easily PID design problems such as: finding the controller that optimizes the performance while fulfilling a prescribed robustness constraint and a maximum permitted control effort due to noise. The tuning procedure has been implemented in a software tool that can be freely downloaded from https://sites.google.com/a/uji.es/freepidtools/. The tool allows to calculate the controller in continuous time (both system and controller are continuous-time models), or in digital mode, using the ZOH discrete system model and a discretized PID. The robustness can be defined in three different ways, involving the phase margin, the gain margin and the sensitivity peak. The performance can be defined in several ways, including the IAE, ITAE, ISE and other. The control effort due to measurement noise is computed as the high-frequency gain of the controller in the case of a continuous-time controller design or an accurate metric evaluated at the sampling rate in the case of digital (sampled data) controller design. Some examples show the validity of the approach and the use of the tool.

Keywords: PID controller, PID tuning, CACSD

1 Introduction

A wide research area in automatic control is the one devoted to the tuning of PID controllers. Many books and papers have been published about this subject. There are academic works, that deal with theoretic approaches, and more practical ones that try to give simple tuning methods to be applied at the plant floor by engineers. In this sense, we can differentiate between experimental tuning procedures, that assume no model of the plant is available, and model-based ones, that assume that a suitable model of the plant is known. This work focuses in the design of PID controllers assuming that a continuous time transfer function model of the plant to be controlled is available.

The design of PID controllers has been tackled in several classical books about control theory, like (Ogata 2003, D'azzo and Houpis 1966, Dorf and Bishop 2007, Phillips and Harbor 1999, Franklin et al. 2005). In those books, the authors propose simple methods based on guaranteeing a prescribed phase margin. The main idea is to fix the controller integral (or lag) zero at a value between 1/8 and 1/10 of the final gain crossover frequency. However, in most cases, this idea leads to a low value of the integral gain, resulting in a poor performance (measured for example by a large IAE). One can find some commercial tools that can help the designer to compute the controller parameters.

In order to achieve better performance in the controlled plant, several works propose a PID tuning through a direct numerical optimization of a given performance index (usually the integral of the absolute error, IAE, of the disturbance response), constrained to some robustness conditions. Among these works, one can cite (Liu and Daley 1999, Hwang and Hsiao 2002, Toscano 2005, Astrom et al. 1998, H. Panagopoulos and T.Hagglund 2002). We can find translations of the PID optimal control design to optimization problems of different nature. Some works rely on iterations over convex optimization problems (Boyd et al. 2016, Mercader et al. 2016, Hast et al. 2013).

In recent times, the development of metaheuristic optimization methods has led to the proposal of several PID tuning strategies that are based on the solution of different constrained optimization problems. Those optimization problems are non linear and non convex, and methods based on evolutive algorithms, particle swarm optimization, artificial bee colony, gray wolf and other, have been applied successfully to the PID tuning problem. In Reynoso-Meza et al. (2013), a survey of the application of evolutive algorithms to PID tuning is presented. In several recent works, as Levy et al. (2019), Zhao et al. (2011), Amirinejad et al. (2014) or Romasevych et al. (2020), some metaheuristic methods, as particle swarm or bee colony optimization are used to find the optimal parameters of PID controllers.

The drawback is that the numerical optimization tends to be very complex (several local minima could exist), and no tools are offered to solve directly those optimization problems for such a common problem as PID control design. Furthermore, as PID control is a widespread control technique, it happens that the need to solve an optimization problem, can be a bottleneck to extend optimal PID control design.

A different approach, that can be found in (Ho et al. 1998, Madhuranthakam et al. 2008, Toscano 2005, Tavakoli et al. 2005), relies on approximating the process to be controlled by a simple first-order with a time delay or a second order with time delay model, and proposes an approximate optimization based on those simple models. The result of those proposals are usually a set of simple tuning equations, but the drawback is that those simple models are not always sufficiently accurate for approximating the behavior of the process.

Therefore, we find tools whose use can be easily extended but that achieve poor performance, and advanced design techniques for which we do not find tools that help engineers to find those optimal controller parameters. In this sense, this work is motivated by the fact of designing PID controllers that fulfill several requirements as the ones included in the advanced design techniques, but that require a computation that can be solved in a reasonable time and with a simple optimization procedure. Furthermore, this work is also motivated by the need to offer

a simple tool that helps engineers to easily find the required controller. In that sense, we look for a tool that has as inputs the model of the plant, the requirements on performance (in terms of disturbance rejection and control effort) and in robustness (to face model uncertainties), and that gives as an output the optimal controller parameters and simulations of the expected closed loop behavior.

Other authors have developed interactive tools for PID tuning, especially with educational purposes. For example, in (Garrido et al. 2018), an interactive tool is presented for PID tuning in the frequency domain. Its main purpose is educational. It includes an interactive 2 dimensions graph in the parameter space that shows the curves and regions defined by different robustness specifications (as gain margin, phase margin or maximum of the sensitivity and complementary sensitivity functions). It allows the user to design the controller by simply selecting one point in the parameter space, and to compare the response of different controllers in simulation. In (Guzmán et al. 2008), three interactive tools for PID tuning and analysis, with educational purposes, are presented. The first one is for PID basics, where a manual PID tuning is performed. The second one is for loop shaping tuning, where the user can select the constraints in terms of robustness (gain margin, phase margin or maximum sensitivity). In the case of PI, the user can select the frequency where the constraint is to be applied. In the case of PID, the user can also select the slope of the Nyquist curve in the selected frequency point. As a difference with the proposed tool, the previous tools do not include tuning rules for the derivative filter parameter, therefore, it is not easy to find the compromise between performance and control effort due to noise. Furthermore, those tools do not solve any optimization problem (for example minimizing IAE), despite they can be used to solve manually the problem of maximizing integral gain. In (Díaz et al. 2017) an interactive tool for loop shaping based controller design is presented, that includes the PID with derivative filter as a particular case. This tool is more complete than the previous one, allowing to define different constraints to be fulfilled in the frequency response of the open loop transfer function, showing graphically those constraints. However, the controller tuning must be made manually, in an iterative way. This makes the tool very interesting for educational purposes, to understand the effect of each controller parameter on the frequency response and the loop behavior, but very hard to be used to find an optimal compromise between performance and noise amplification.

With the aim of solving an advanced control design with simple optimization techniques, in (Sanchis et al. 2010), the proposed idea is to use the quotient between gain crossover frequency and controller zero as a dimensionless tuning parameter, that allows developing a simple procedure for optimal PID tuning. The procedure allows us to obtain the PID that minimizes the IAE fulfilling a robustness constraint, defined in terms of an exact required phase margin, and a lower bound in the gain margin. The resulting optimization problem can be easily solved numerically thanks to the single tuning parameter proposed.

In this paper we propose a PID tuning strategy that roots on that work, and is based on the concept of fixed robustness tuning line. In this line, each point is a PID controller that fulfills the required robustness requirement. The central point of the line is the PI controller that optimizes the performance while fulfilling the robustness requirement. The left part of the line is formed by PI controllers that are slower than the optimum, and produce a lower noise amplification. The right part of the line is formed by PID controllers with increasing values of derivative filter parameter, N, that optimize the performance while fulfilling the robustness constraint. The tuning line is monotonically increasing in terms of performance and noise amplification. This tuning line has no discontinuities, since a PID with $N = 0$ is a PI controller. Once the robustness has been selected, the PID tuning reduces to select the point in the tuning line that results in the required compromise between performance and noise amplification. For example, to obtain the controller that optimizes performance while keeping the noise amplification below a prescribed level, we have to select the point in the tuning line that produces exactly that noise amplification.

The previous tuning strategy is not really useful without a tool that implements it in practice.

In this paper, we describe a free software tool to design PID controllers following the previous strategy. The application allows the user to select the required robustness, and to select any PID controller in the tuning line with a simple click of the mouse. Therefore, it solves, in an easy way, PID tuning problems where the objective is finding the optimal compromise between performance and noise amplification, while fulfilling a robustness constraint.

The layout of the paper is as follows: first, the PID design problem is stated. Then, the controller tuning procedure is described, and the basics of the calculation method are explained in detail. After that, the Java based PID design tool is described, and several design examples are developed to show the validity of the tuning procedure and the application. Finally the conclusions are summarized.

2 Problem statement

The SISO PID control loop considered in this paper is shown in figure 1, where r is the reference signal to be tracked, u is the control action, y is the controlled output, d is an input disturbance and v is the measurement noise.

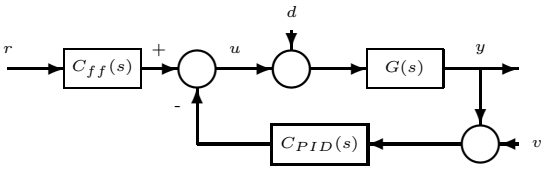

Figure 1. SISO PID control loop

The PID controller is assumed to have the following structure

$$C_{PID}(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{N} s} \right) \tag{1}$$

$$C_{ff}(s) = K_p \left(b + \frac{1}{T_i s} + \frac{c\, T_d s}{1 + \frac{T_d}{N} s} \right) \tag{2}$$

The tuning of the PID controller implies the selection of the most adequate values of all the parameters: K_p, T_i, T_d, N, b, and c. We assume that a model of the plant is known in the form of a transfer function $G(s)$ (obtained for example through identification). The proposed tuning procedure takes into account three factors that describe the behavior of the controlled system: robustness, performance (speed response) and control effort due to high frequency noise. Basically, the user chooses the desired robustness, and the tuning procedure facilitates to find the optimal compromise between performance (speed response) and control effort due to noise, while fulfilling the robustness constraint. For example, one can calculate the controller that optimizes the performance while fulfilling the required robustness and an upper bound in the control effort due to noise. Or one can calculate the controller that minimizes the control effort due to noise while fulfilling the required robustness and a required performance. Or the third option: the controller that maximizes robustness while achieves a required performance and an upper bound in the control effort due to noise. The three factors in the PID design are described in the sequel.

2.1 *Robustness*

We propose to use three possible alternative ways to define the desired robustness to be imposed in the design:

(1) Exact required phase margin and lower bound in the gain margin ($\Phi_m = \Phi_{m,req}$, $G_m \leq G_{m,req}$). An exact phase margin is imposed, but the gain margin is only constrained as a lower bound. A prescribed phase margin can not be used alone to guarantee robustness, as it is well known. To force additionally a prescribed exact gain margin may result in an unnecessary constraint that reduces performance, especially in systems without delay. The lower bound in the gain margin guarantees the robustness while do not impose an additional constraint if not needed.
(2) Exact required phase margin and upper bound in the sensitivity peak ($\Phi_m = \Phi_{m,req}$, $M_s \geq M_{s,req}$). Another way of complementing an exact phase margin to guarantee robustness is to use an upper bound in the sensitivity peak. As in the case of the gain margin, the bound does not impose an additional constraint if not necessary.
(3) Exact required sensitivity peak ($M_s = M_{s,req}$). The maximum of the sensitivity function is a single parameter that can be used alone to define the required robustness, as it is well known, because it guarantees a minimum distance of the Nyquist plot with respect the -1 point.

The proposed tuning procedure and the developed tuning tool allows the user to select which of the three options is used to define the desired robustness in the PID design. The tool calculates controllers that fulfill the prescribed robustness.

2.2 *Performance (speed response)*

The performance of the controlled system in terms of speed response can be measured with different metrics. Some of them are related to the response to a step change in the disturbance input, while other are related to the response to a step change in the reference signal. Related to the disturbance response, the most common metrics are the Integral of Absolute Error (IAE), Integral of Time times Absolute Error (ITAE) or Integral of Squared Error (ISE). Related to the reference response, we can also have the IAE, ITAE or ISE, but also the settling time, or the velocity error (the steady state error to unit ramp reference, that is inversely proportional to the integral gain $K_i = \frac{K_p}{T_i}$).

The proposed PID tuning procedure, and the developed tuning tool, allows to deal with different performance metrics, as:

(1) K_i. The integral gain $K_i = \frac{K_p}{T_i}$ is the inverse of the disturbance Integral of Error (IE). If the response is not too oscillatory, this is an approximation of the IAE. On the other hand, K_i is inversely proportional to the velocity error, therefore, maximizing K_i implies minimizing the velocity error.
(2) Disturbance IAE. The integral of absolute error in the step disturbance response is maybe the most commonly used performance measure in the PID design methods. Minimizing the IAE implies to find a reasonable compromise between maximum transient error and time to recover from disturbance. This is the most widely used performance measure in literature.
(3) Disturbance ITAE. The integral of absolute error multiplied by time puts extra weights in the errors that remain in time. Therefore, minimizing the ITAE results in a controller with a higher maximum transient error, but a lower recovery time (if compared to minimizing IAE).
(4) Disturbance ISE. The integral of squared error puts more weight in the higher values of the error, hence minimizing the ISE results in a lower maximum transient error, but a higher recovery time (if compared to minimizing IAE).

(5) Gain crossover frequency, ω_g. Is the frequency where the phase margin is measured, and is related to the speed response (the higher the frequency, the faster the response).
(6) Settling time, t_s. Is the time taken to the output to reach the 98% of the final value with a step reference, remaining in the 2% error band. If the closed loop is sufficiently robust, such that the response is not too oscillatory, the settling time can be approximated as an inverse function of the gain crossover frequency (the higher the frequency, the lower the settling time). Equation (3) shows an approximation of this function obtained with the simplifying assumption that the closed loop is a pure second order model (two poles and no zeros).

$$t_s \approx \frac{8\tan(90-\Phi_m)}{\omega_g} \tag{3}$$

Some of these performance metrics are well suited to be the objective of an optimization problem (for example IAE, ITAE, ISE or K_i), while other are not so good for that purpose (as ω_g or t_s). For example, the settling time, depending on the oscillations, can have a non uniform behavior (can have significant jumps with small changes in the PID parameters), hence it is not a good metric for optimization. The gain crossover frequency, ω_g, is a performance metric that can be used to fix a required value, but not as the objective of the optimization, because if we try to maximize it, the result is an infinite integral time (and thus an infinite IAE). On the other hand, the settling time can be approximated as an inverse function of the gain crossover frequency. Therefore, to fix a desired settling time, an approximate required ω_g can be computed, and then, the PID can be tuned to reach that value of ω_g.

2.3 *Control effort due to noise*

We assume that the measurement noise is a zero average high frequency signal. If the controller is designed in continuous time, then the control effort due to the measurement noise (the peak to peak amplitude of control action fluctuation produced by the measurement noise) depends on the high frequency gain of the controller, as the high frequency gain of the plant is zero. In that case, if the peak to peak amplitude of measurement noise is ν, the peak to peak amplitude of control effort due to noise is

$$u_{noi} = C_{PID}(s=\infty)\nu = K_p(1+N)\nu \tag{4}$$

for PID controller, or

$$u_{noi} = C_{PID}(s=\infty)\nu = K_p\nu \tag{5}$$

for PI controller.

In the case of the PID controller, the control effort due to noise depends on gain K_p, but also on the derivative filter coefficient N. If we want to find an optimum compromise between control effort and performance, the adequate selection of parameter N is crucial.

If the calculation of the controller is done in sampled data mode (using the ZOH discrete system model and a discrete PID), then the previous equation for control effort due to noise is non sense. Assuming a white measurement noise of zero mean, $v(k)$, a possible measure of the control effort due to noise is the standard deviation, that can be computed as:

$$\sigma_u = \sqrt{E[u_{noi}(k)^2]} = \sqrt{\frac{T}{\pi}\int_0^{\pi/T}\left|\frac{C_{disc}(e^{j\omega T})}{1+C_{disc}(e^{j\omega T})G_{ZOH}(e^{j\omega T})}\right|^2 d\omega}\cdot\ \sigma_v \tag{6}$$

where T is the sampling period, $G_{ZOH}(z)$ is the zero order hold discrete equivalent of the plant model, $C_{disc}(z)$ is the discrete PID transfer function, and $\sigma_v = \sqrt{E[v(k)^2]}$.

3 Controller tuning procedure

The basis of the procedure for PID tuning is the concept of fixed robustness tuning line. Figure 2 shows a representation of this line. Each point of this virtual tuning line represents a controller with the optimal compromise between performance and control effort due to noise, fulfilling a predefined robustness constraint. For defining the line, a performance index must be chosen (as for example the disturbance IAE), and a robustness definition criteria must also be selected (as for example $\Phi_m = \Phi_{m,req}$, $G_m \leq G_{m,req}$). The central point of the tuning line is the PI controller that optimizes the performance (minimizes IAE for example), fulfilling the robustness constraint. That point achieves certain value of IAE, and certain control effort due to noise ($K_p\nu$ in a continuous time design, or (6) in a sampled time design). The points on the left side of the line are PI controllers that are slower than the optimum PI (the one with minimum IAE) and with a lower control effort due to noise. The points on the right side of the line are PID controllers, with increasing values of derivative filter N, that optimize the performance for the given robustness. In fact the optimal PI is equivalent to the optimal PID with $N = 0$, and hence, the tuning line has no discontinuities. As we move to the right of the line, increasing N, the performance index (IAE) decreases, while the control effort due to noise increases. Figure 3 shows the relation between performance ($1/IAE$) and control effort due to noise, for different tuning lines obtained for different robustness, for process (13). The points that correspond with the optimum PI controllers are shown in circles joined by a dotted line. It is important to highlight that the plots are monotonously increasing (the more control effort, the higher performance), as required for the tuning line concept to be valid.

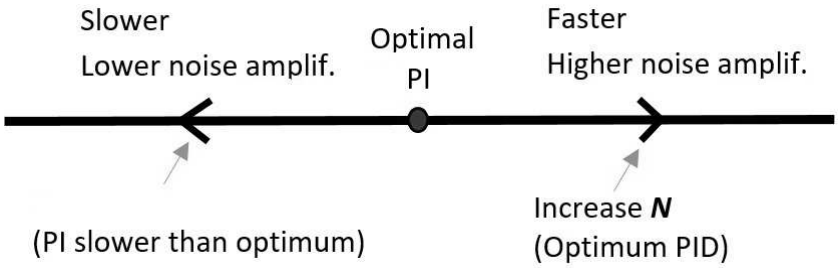

Figure 2. Tuning line for fixed robustness.

The proposed PID tuning approach simply consists of fixing the required robustness, and then selecting from the tuning line, the point with the desired compromise between performance and control effort due to noise. The proposed procedure is to first choose the central point of the line (i.e. select the PI that optimizes performance), and then, decide if we want to reduce performance and control effort due to noise (choosing one point on the left), o we want to increase performance and control effort due to noise (choosing one point on the right).

Taking into account the three factors that play a role in the controller tuning (robustness, performance and control effort due to noise), three tuning problems can be formulated:

(1) Obtain the controller that optimizes performance while fulfilling a robustness constraint and an upper bound in the control effort due to noise. This tuning problem is solved by simply selecting the point in the tuning line that has the maximum permitted control effort due to noise.
(2) Obtain the controller that minimizes control effort due to noise while fulfilling a robustness constraint and an upper bound in the performance index. This tuning problem is solved by simply selecting the point in the tuning line that has the required value of the performance index.
(3) Obtain the controller that maximizes robustness while fulfilling an upper bound in the control effort due to noise, and an upper bound in the performance index. This tuning

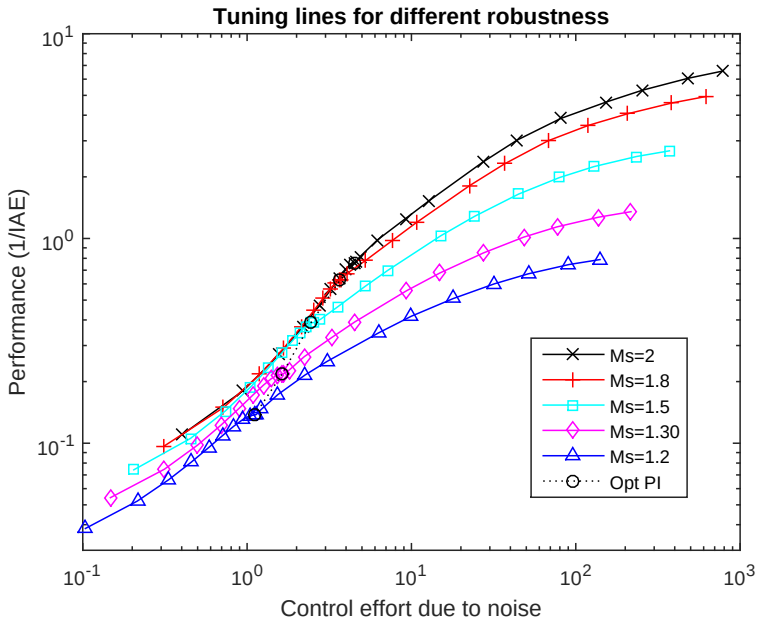

Figure 3. Performance versus control effort due to noise in tuning lines of different robustness.

problem is solved by an iterative approach: select a high robustness and solve problem 1 or problem 2. If the resulting controller does not fulfill either the control effort due to noise or the performance, decrease the robustness and repeat the calculation, until both requirements are fulfilled.

The PID tuning tool that has been developed allows to select the desired robustness and to easily move along the tuning line, computing automatically the controller parameters that correspond to the selected point. Therefore, it allows to solve the three tuning problems in a straightforward manner.

When the calculation of the controller is done in sampled data mode (using the ZOH discrete equivalent of the plant and a discrete PID), the tuning line is different for each sampling rate, but the idea still applies. Figure 4 shows the relation between performance ($1/IAE$) and control effort due to noise, for tuning lines obtained for different sampling rates with the same robustness (defined as a required value of M_s), for process (13). The points that correspond with the optimum PI controllers are shown in circles joined by a dotted line. For all the sampling rates, the plots are monotonously increasing, as required by the tuning line concept. It is interesting to note that the ratio performance-control effort always worsens as the period increases. Furthermore, the range of possible controllers is more limited as the period increases. This is specially true for the PID part of the tuning line, that for large periods reduces almost to a point (as shown for example in the T=10 case). This means that, as the sampling period increases, the increase of derivative filter parameter N produces a lower increase on the performance and control effort due to noise. On the other hand, once the required robustness has been chosen, the selection of the sampling rate for implementing the PID controller affects the compromise between performance and control effort due to noise that can be achieved. If a given performance and a given control effort are required, i.e., a point in the plot is fixed, then the required sampling period would be the one whose tuning line includes that point. In practice, it could be computed by iterating different tuning lines (of different periods) until the required performance and control effort are exactly met.

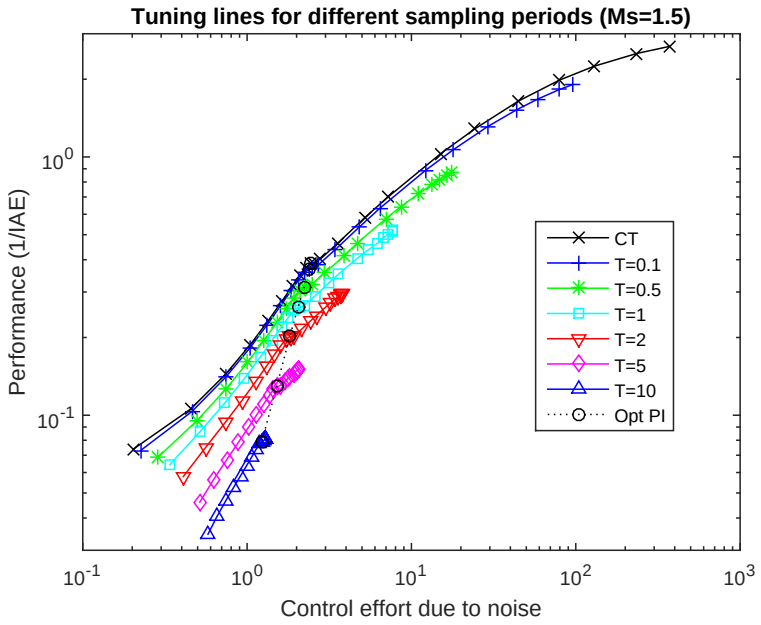

Figure 4. Performance versus control effort due to noise in tuning lines for different sampling rates.

4 Controller calculation basics

The implementation of the fixed robustness tuning line on a software tool requires in practice the solution of several optimization problems. The basis of the proposed algorithm to solve those optimization problems (implemented in the internal calculations performed in the application) is the use of a dimensionless tuning parameter, $a = T_i\omega_g$, where T_i (the integral time), and ω_g (cross over frequency) are initially unknown. In the case of PI controllers, if the value of a is fixed, the phase provided by the controller at the frequency ω_g (where the phase margin is measured) is known (this phase, $\arg(1+\frac{1}{aj})$, only depends on a). This allows to calculate easily the PI controller that achieves a certain desired phase margin. We only need to find the frequency ω_g where the process has the needed phase: $\arg(G(j\omega_g)) = -180 + \Phi_m - \arg(1+\frac{1}{aj})$. Once ω_g is known, T_i can be obtained as $T_i = a/\omega_g$, and finally K_p is computed by imposing that the magnitude at that frequency is 1: $|C(j\omega_g)G(j\omega_g)| = 1$, i.e. $K_p = \left|(1+\frac{1}{aj})G(j\omega_g)\right|^{-1}$.

In order to select the best value of parameter a we propose to optimize the chosen performance index. This is a one dimension optimization problem, easy to be solved numerically. The PI controller (K_p and T_i) would then be the solution of the following optimization problem:

$$\min_a IAE \qquad (7)$$
$$s.t.$$
$$\Phi_m = \Phi_{m,req} \quad ; \quad G_m \leq G_{m,req}$$

As parameter a is dimensionless, this problem can be easily solved in a very short time, iterating with different values of a in a known range ($a \in [0.1, 10]$), computing for each of them the controller that achieves the required phase margin. If the robustness is defined in terms of M_s instead of Φ_m, the optimization problem would be formulated as:

$$\min_{a} IAE \quad (8)$$
$$s.t.$$
$$M_s = M_{s,req}$$

The computation to solve this problem still relies on finding the optimum a for a given phase margin, and then iterating the phase margin until the desired M_s is achieved. Thus, the computation time when the robustness is defined by M_s is sensibly higher, because a double iteration is performed (in Φ_m and in a). However, the computation time is still reasonable for an interactive tuning tool implementation.

In the case of the PID controller the previous ideas also apply if, in addition, the parameter of the derivative filter, N, and the ratio between the integral and derivative times, T_i/T_d are fixed a priory. In that case, if the user selects the value of a, the controller that achieves the desired phase margin can be computed automatically following the same procedure as the PI controller. Again, the selection of the best value for a would be the result of an easy to solve single parameter optimization. Therefore, the PID controller (K_p, T_i, T_d and N) would then be the solution of the following optimization problem:

$$\min_{a} IAE \quad (9)$$
$$s.t.$$
$$N = N_{req} \quad ; \quad \frac{T_i}{T_d} = \beta$$
$$\Phi_m = \Phi_{m,req} \quad ; \quad G_m \leq G_{m,req}$$

where N_{req} and β are constant values that are fixed a priory. In the case of the PID, the optimization problem can be extended to find the optimal value of the ratio T_i/T_d. This results in a two dimensional optimization problem instead a one dimensional one, but the computational cost is still reasonable, because T_i/T_d is also a dimensionless parameter with a known range ($T_i/T_d \in [0.5, 20]$). The formulation of the problem to be solved to find the PID would be:

$$\min_{a, \frac{T_i}{T_d}} IAE \quad (10)$$
$$s.t.$$
$$N = N_{req} \quad ; \quad \Phi_m = \Phi_{m,req} \quad ; \quad G_m \leq G_{m,req}$$

This problem can be solved performing iterations on solving problem (9) for different values of $\frac{T_i}{T_d}$. As explained in the PI case, if the robustness is defined in terms of M_s instead of Φ_m, the optimization problem would be formulated as:

$$\min_{a} IAE \quad (11)$$
$$s.t.$$
$$N = N_{req} \quad ; \quad \frac{T_i}{T_d} = \beta \quad ; \quad M_s = M_{s,req}$$

or

$$\min_{a,\frac{T_i}{T_d}} IAE \tag{12}$$

$$s.t.$$

$$N = N_{req} \quad ; \quad M_s = M_{s,req}$$

Again, the computation still relies on solving (9) or (10) and iterating the phase margin until the desired M_s is achieved. Thus, the computation time when the robustness is defined by M_s is sensibly higher, because a double iteration is performed in the case of problem (11) (in Φ_m and a), and a triple iteration in case of (12) (in Φ_m, a and T_i/T_d). However, the computation time is still reasonable to be implemented in an interactive PID design tool.

The developed application implements the previous ideas: it allows the user to select the value of a, computing automatically the controller that achieves the desired Φ_m or the desired M_s (for a PI o for a PID with the selected values of N and T_i/T_d), or it solves the previous optimization problems (the one selected by the user) with the click of a button.

The idea of the fixed robustness tuning line is implemented in the following way. The user selects the desired robustness and the performance index to be used in the optimization. Then the optimum PI controller is computed solving the problem (7) or (8). This PI controller is the central point of the tuning line. For less aggressive responses, we simply reduce the value of the parameter a, leading to slower PI controllers with the same robustness. On the other hand, if we want a more aggressive response, we change the controller to be a PID, we choose a value for the derivative filter N, and compute the optimal PID controller. To do so, the application solves one of the optimization problems described previously for PID, depending on the selection of the ratio T_i/T_d (fixed or arbitrary). By increasing N we travel along the tuning line for more aggressive controllers. The optimization of T_i/T_d leads in general to a better performance-control effort ratio than using a fixed T_i/T_d, but the resulting controller is usually less robust, as it will be shown in the examples. For example, if the robustness constraint is defined as a required value of M_s, despite the same value of M_s is reached, the resulting values of Φ_m or G_m are lower, and the response more oscillatory. Therefore, the optimization of T_i/T_d is not necessarily the best choice, and often, a fixed value is used instead (for example $T_i/T_d = 4$).

In the application developed to implement the tuning procedure, the user decides whether T_i/T_d is fixed or should be optimized, and the application solves automatically the selected optimization problem for the chosen value of N. In any case, the selection of a controller from the tuning line is straightforward.

The proposed approach requires the solution of several optimization problems that could be unfeasible. In fact, there is no guarantee that those optimization problems have feasible solutions for all general process models. If we take, for example, a double integrator ($G = 1/s^2$), and a requirement of phase margin, as the phase of the plant is constant (-180^o), if we use a PI controller (that has negative phase), it is impossible to reach any positive phase margin, hence, no matter which phase margin we choose, the problem will be unfeasible. When the problem is infeasible, the application leads to an arbitrary controller, that does not fulfill the robustness constraints. However, we have tested a very wide batch of models (for example those included in (Astrom et al. 1998)), and the problems are feasible for the most common robustness constraints for all the models.

5 Java based design tool

An application has been developed in Java to implement the PID design strategy and to simulate the response of the controlled system to setpoint and disturbance changes. The tool can be downloaded for free from the site https://sites.google.com/a/uji.es/freepidtools/ , where other

free tools can also be found (for identification for example). The tool is composed by a single executable java file (ejsPIDmodelbased2020.jar). As other java applications, it can be run in different platforms. The only requirement is to install the java runtime, that can be downloaded from https://www.java.com/en/download/. Figure 5 shows the starting window of the application, where the model of the system is defined. There are two ways for defining the model: introducing the coefficients of numerator and denominator of the continuous time transfer function (up to order 6), or introducing the gain and time constants, selecting the terms that are part of the model. The model can have up to 3 real poles, one pair of complex poles, one real zero, one integrator, and time delay. In order to use the sampled data PID design mode, the model must be defined with time constants. The model can be saved to a file to be loaded afterwards. In this window, the maximum and minimum values that define the control action saturation are also defined, and the amplitude of the sensor measurement noise.

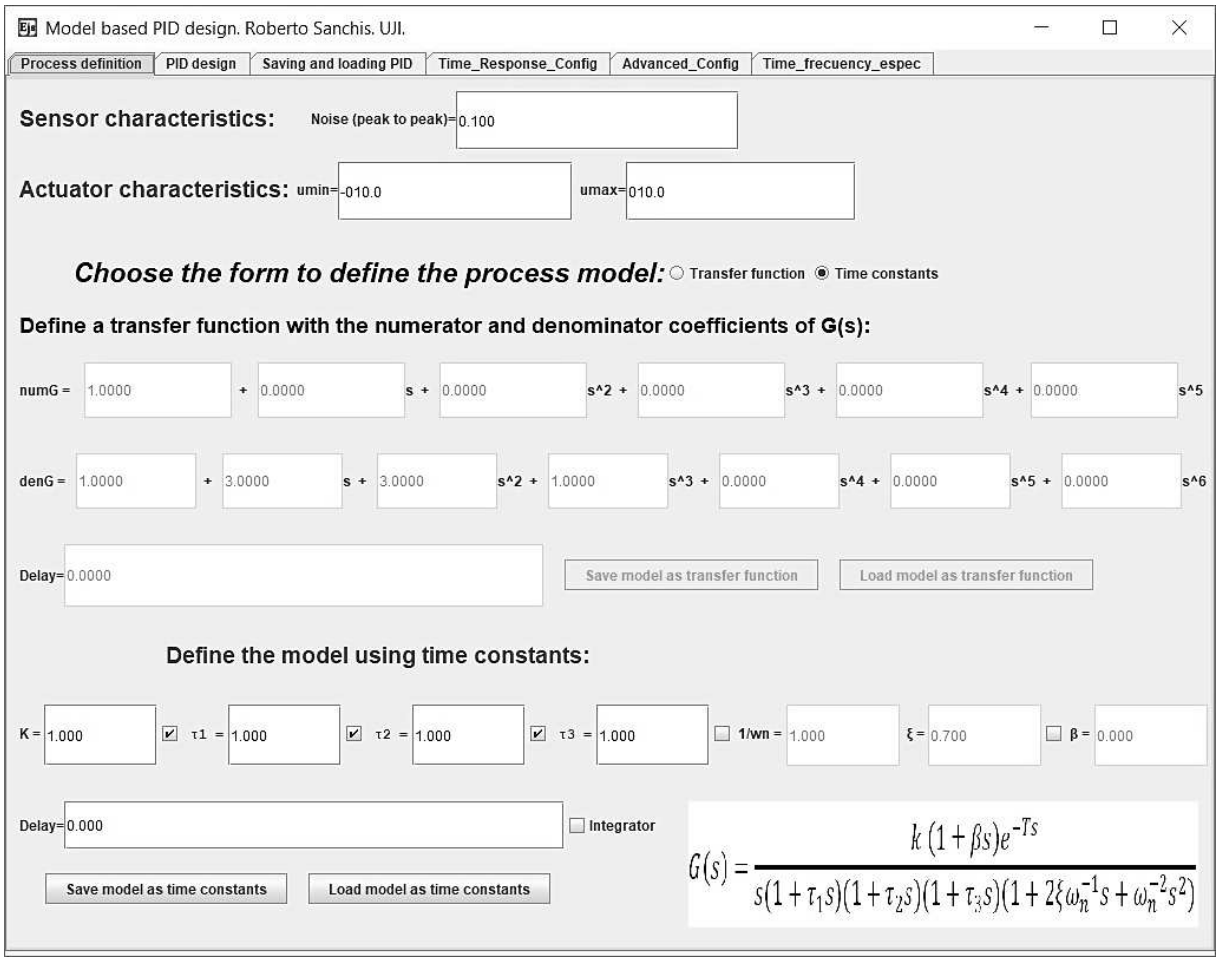

Figure 5. Java application: process model definition window.

Figure 6 shows the main PID design window of the application. The main graph shows the Nyquist diagram of process plus controller, showing graphically the robustness margins. The left part of the window has three tabs, each one for a different way of designing the controller. The "Manual PID" tab is for introducing manually each one of the controller parameters, K_p, T_i, T_d and N in ISA PID format, or K_p, K_i, K_d and τ_d in parallel format. The tab "Auto PID" allows the design of the PID following the procedure described in the previous sections. The procedure to design the controller can be summarized as:

- Select the PI type of controller.
- Select the way of defining the robustness among the three options.

- Select the performance index for optimization (default IAE).
- Use the sliders to fix the desired robustness. If we select one of the phase margin options, the application will propose automatically the bound of the other parameter (either gain margin or sensitivity peak). The user can change manually the proposed value if desired, unchecking the corresponding check box.
- Press the "Opt" button next to the a parameter slider. This finds the PI controller that optimizes the performance index, fulfilling the required robustness. This is the central point of the tuning line. The performance index and the control effort due to noise are shown above the a slider to guide the next decision: either keep that PI controller, or design a slower one with a lower control effort due to noise, or design a faster PID with a higher control effort due to noise.
- If the user wants a slower design, then simply reduce the value of parameter a with the slider, to find a slower PI with the desired compromise between performance and control effort due to noise.
- If, on the other hand, the user wants a faster response, then select type of controller PID, check the box "Aut" below the "Opt" button, next to the a slider, and select if he wants a fixed value of T_i/T_d for the tuning line (checking the box "Fix Ti/Td"), or he wants the application to compute the optimum T_i/T_d (unchecking the box "Fix Ti/Td"). Then, every time the N value is changed with the slider, the application calculates the optimum controller, hence moving along the right part of the tuning line is as simple as changing the N slider value to find the desired compromise between performance and control effort due to noise. If the box "Fix Ti/Td" is unchecked, the algorithm is computationally more complex and, hence, the change in N to find the desired controller must be done with patience. This is especially true if the robustness index selected is M_s. To reduce the computation time, a check box has been added ("Fast approximate calculation"), to reduce the number of points and the accuracy used for the calculation. This box is checked automatically when the box "Fix Ti/Td" is unchecked.

On the other hand, the tab "Tuning line" allows the design of the controller directly applying the tuning line concept described in the previous section. The robustness and performance index selection are the same, and also the selection of the ratio T_i/T_d (fixed or optimized). However, the user does not select the type of controller (PI or PID), nor the value of derivative filter parameter N. Instead, the user simply selects one point of the tuning line. This line has been implemented as a slider (defined with an auxiliary variable γ), with some buttons (xL, mL, L, M, R, mR, xR) that can be used to select predefined points of the line (see figure 7). For example, the central button (M) selects the optimal PI controller, that is the central point of the tuning line (that also corresponds to $\gamma = 0$). The buttons on the left (L, mL and xL, that correspond to $\gamma = -4$, $\gamma = -7$ and $\gamma = -10$) define slower PI controllers, while the buttons on the right (R, mR and xR, that correspond to $\gamma = 2.5$, $\gamma = 5$ and $\gamma = 10$) define faster PID controllers of increasing N. One can select intermediate points by dragging the slider or using the buttons "+" or "-". When the box "Fix Ti/Td" is unchecked, the application computes the optimal value of T_i/T_d for each point of the tuning line on the right of the central point. In that case, the computation of each point is time consuming (especially if the robustness constraint is defined in terms of M_s), hence, the slider is disabled, and only the push buttons are enabled to select one point of the tuning line.

Below the controller design tabs, there are some check boxes. The first one is for reducing the number of points and the accuracy in the calculations to reduce the computation time. The second one is to show in a new window the Bode diagram of process and controller. The third one is to show or hide the time response of the closed-loop system. And finally, the last one ("Digital calculation"), if checked, configures the application to perform all the calculations in sampled data mode, using the provided sampling time. In this case, the ZOH equivalent discrete model of the system is used, with a discrete PID transfer function, to compute the robustness measures. The performance indexes like IAE are computed using a sampled data simulation, taking into account the inter sampling behavior, while the control effort due to noise is computed using

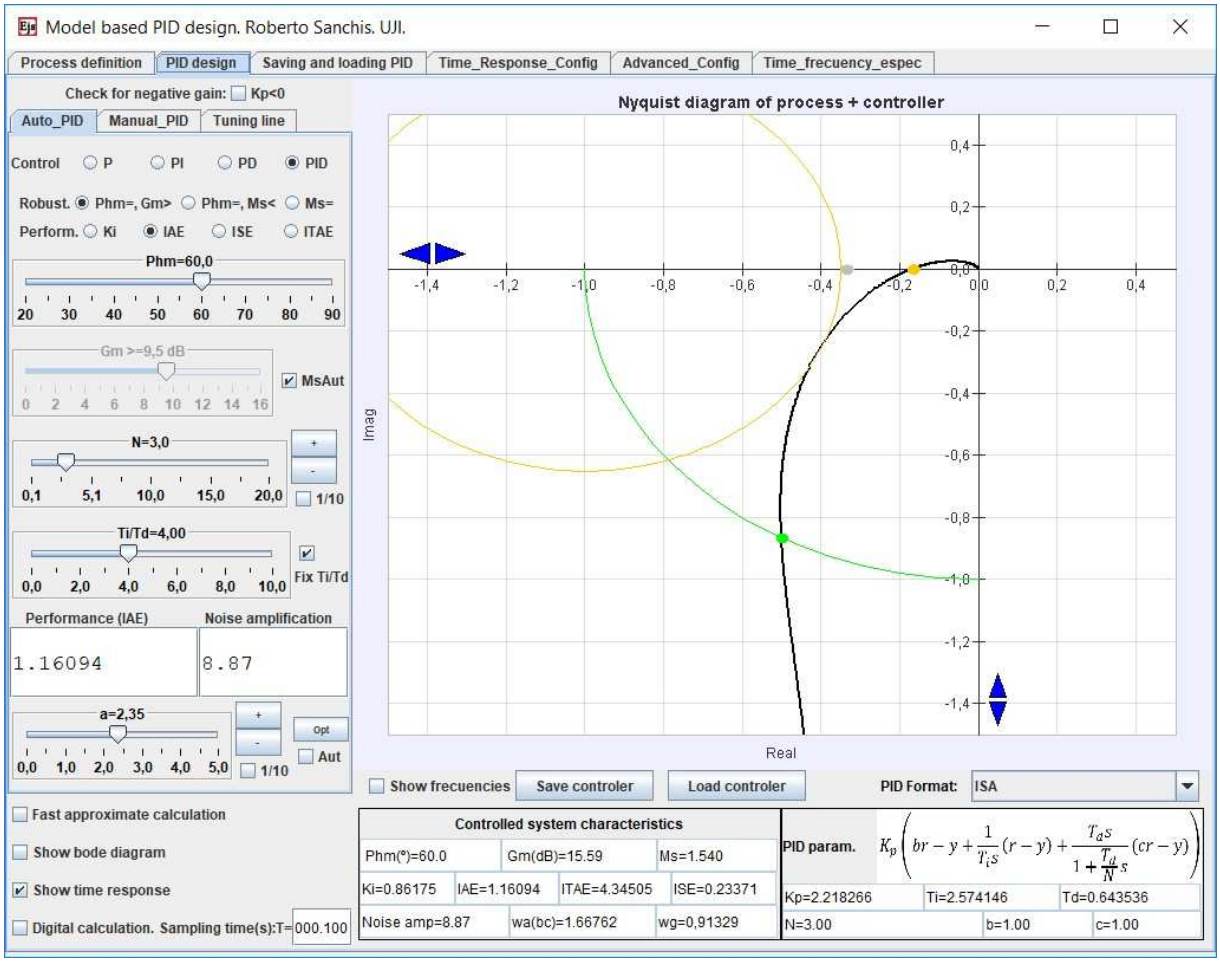

Figure 6. Java application: main PID design window.

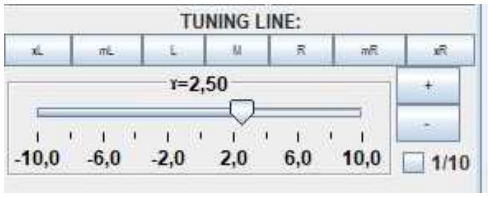

Figure 7. Java application: tuning line implementation.

equation (6). The digital calculation feature is especially interesting when the sampling period for controller implementation is relatively high compared to the time response of the closed loop. In that case, the characteristics of the continuous time design can be quite far from the digital implementation behavior (for example, the true robustness margins can be quite lower than expected, and the system much more oscillatory).

Once the controller has been designed, the application shows the time response of the closed loop. Figure 8 shows the time response window. By default, it shows the response to a step change in the setpoint and, once settled, the response to a step change in the input disturbance. It also shows the control input in another graph. For those type of signals, the tool computes the most common performance indicators. For the reference response: overshoot and settling time (98%). For the disturbance response: IAE, maximum error, and settling time. It also shows the amplitude of control action fluctuation due to noise. The user can decide if the control action saturation is applied or not. In the case of control action saturation, a simple anti-windup method

is implemented, consisting of freezing the integral part when the control action saturates in the same direction of the tracking error (i.e. if $(u > umax) and (e > 0)$ or $(u < umin) and (e < 0)$ the integral part remains unchanged). Another interesting feature is the possibility of performing the simulation in digital mode, with the selected sampling period, to test the effect of the sampling period in the behavior. This is different to the digital controller calculation mode, explained before. The weighting factor for the proportional and derivative terms of the reference, b and c, can also be changed in this window, to test the effect of this parameters over the reference response.

Figure 9 shows a window for configuring the reference and disturbance signals for the time response simulations. By default, both, reference and disturbance are unitary step signals, but this window allows the user to choose the type of signal and the value. Step, ramp or arbitrary signal can be selected. In the case of arbitrary signal, this is defined as a sequence of points (time, value), that are joined by straight lines. The arbitrary signal can be assigned to the reference or to the disturbance, allowing a wide set of possibilities. A typical example is the reference of a thermal system with a heating ramp and a cooling ramp. In the case of selecting ramp reference, instead of step reference, the maximum tracking error and the velocity error are shown in the time response window, instead of the overshoot and settling time. For arbitrary references, the maximum tracking error and the average absolute tracking error are shown instead.

6 Examples

In this section, some examples are developed with the application to illustrate the proposed PID design procedure.

6.1 *Example 1*

Consider a system modeled by the transfer function

$$G_1(s) = \frac{1}{(1+10s)(1+s)^2} \tag{13}$$

Assume that we are interested in minimizing the disturbance IAE, but we want a robustness defined by $\Phi_m = 60^o$, $G_m \geq 9.5dB$, while we want to maintain the high frequency noise amplification below 2. This means that the fluctuation of the control action due to noise will be only two times the amplitude of measurement noise (this could be required, for example, when the actuator has a mechanical wear that should be avoided). Using the application, if we select PI controller, define the required robustness and select the IAE performance index, pushing the "Opt" button, the optimal PI controller is calculated. The noise amplification is 2.72, that is too high, while the IAE is 3.47. Therefore, we need to make the controller less aggressive. For that purpose, we reduce the value of parameter a (using the slider) until the noise amplification is just below 2. The resulting IAE is 3.79. Figure 10 shows the designed controller characteristics. We can see that the gain margin constraint do not affect the design, because it is fulfilled in excess. In the time response window, we have tuned by hand the weighting factor b to reduce the overshoot in the reference response.

Assume now that we want to minimize the IAE, with the same robustness, but we can accept a maximum control effort due to noise of 20 times the measurement noise (because the actuator has no mechanical wear problems). Starting with the optimal PI controller, the noise amplification was 2.72, so, the controller can be made more aggressive. Selecting PID controller, checking box "Aut", and moving the N parameter slider until the noise amplification is just below 20, we get the optimal controller that maintains a ratio $T_i/T_d = 4$, leading to an IAE of 1.03. Figure 11 shows the designed PID controller. In the time response window, we have tuned by hand the weighting factors b and c to reduce the overshoot in the reference response. If the check box

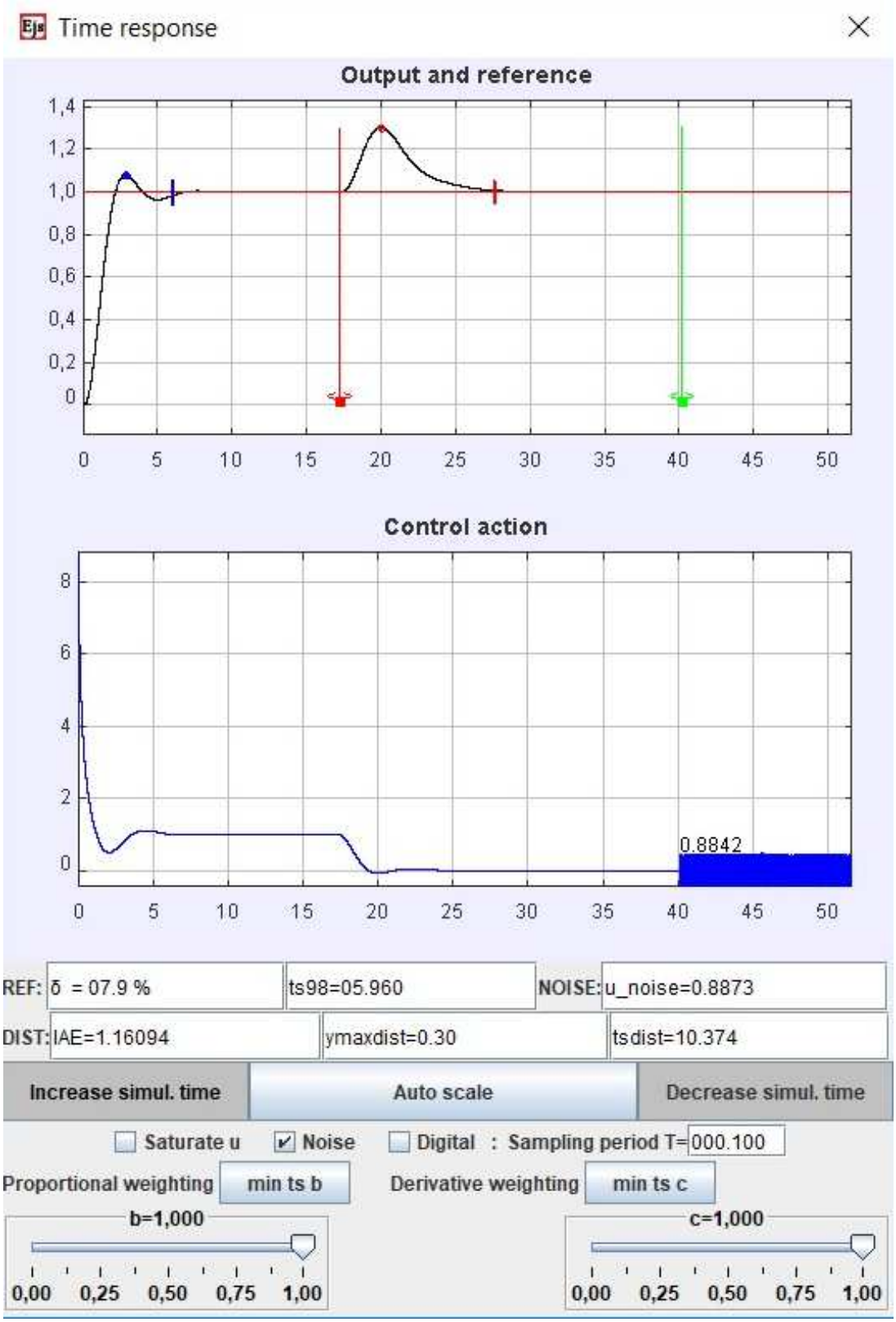

Figure 8. Java application: time response window.

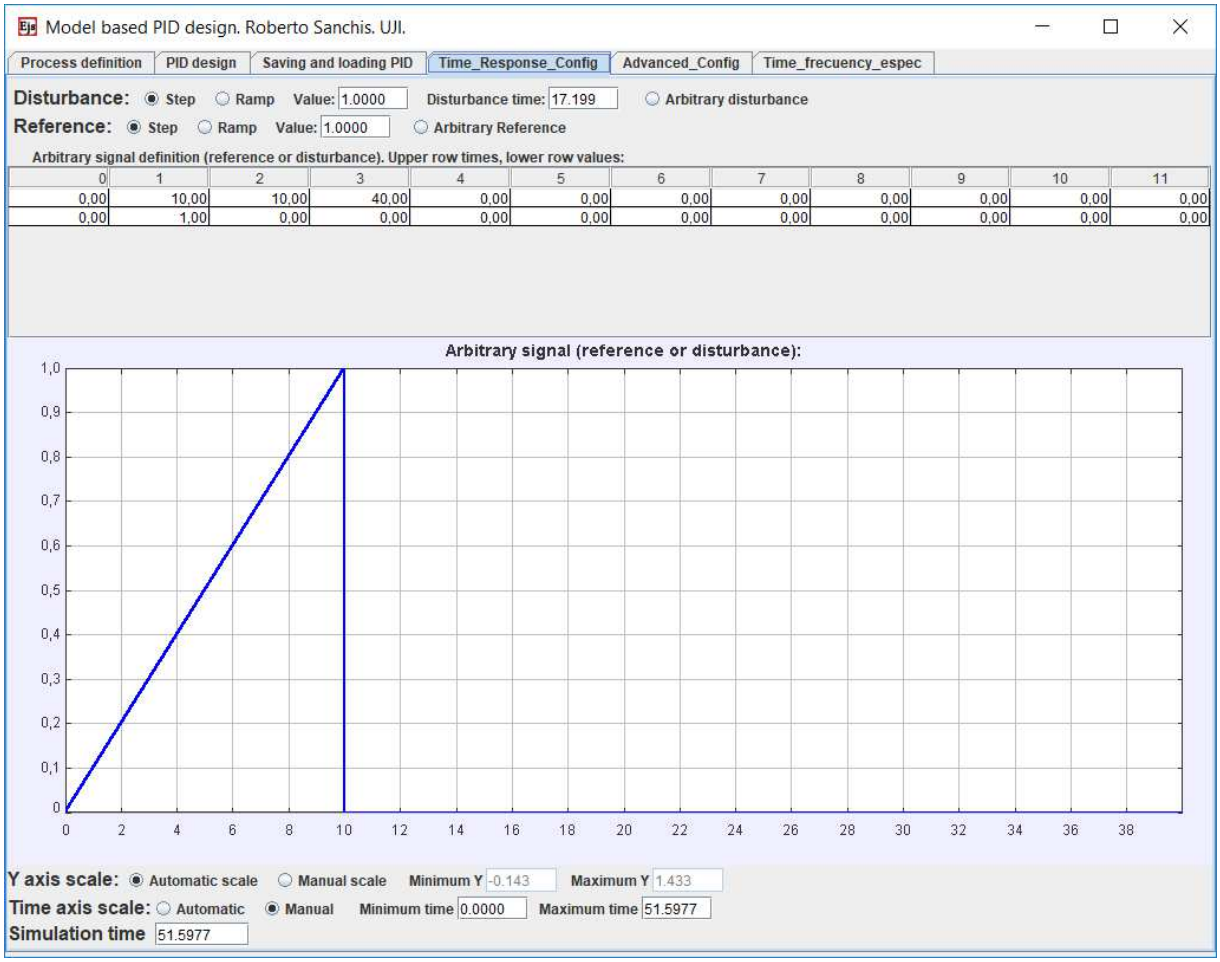

Figure 9. Java application: time response configuration window.

"Fix T_i/T_d" is unchecked, then the application computes the optimum value for the ratio T_i/T_d. Changing N until the noise amplification is just bellow 20 leads to the controller shown in figure 12. The optimal value of T_i/T_d is 2.46. The IAE is now a little bit smaller (0.95), but in fact, there is a cost in term of robustness. The robustness constraint is fulfilled, but the resulting gain margin is lower (and M_s higher) than the controller with $T_i/T_d = 4$. As a result, the time response is more oscillatory, and the reference response is more difficult to be fine tuned with the weighting factors b and c, leading to a higher settling time than in the case $T_i/T_d = 4$.

Consider now that we want to obtain a velocity error below 0.5, so we need $K_i \geq 2$. Assume that the required robustness is now defined by $M_s = 1.5$. If we fix $T_i/T_d = 4$, the optimal controller is shown in figure 13. The noise amplification is 66.3. If the ratio T_i/T_d is optimized (unchecking the box "Fix Ti/Td"), the controller that achieves $K_i = 2$ is shown in figure 14. The noise amplification is much lower, 16.6 instead of 66.3, but the price paid in robustness is very significant, because, despite having the same M_s, the phase and gain margins are lower. The time response is also much more oscillatory, while the disturbance IAE is much higher. Furthermore, the step reference response has a large overshoot that can not be reduced with the weighting factors b and c. This example shows that the optimization of the ratio T_i/T_d is not always the best choice.

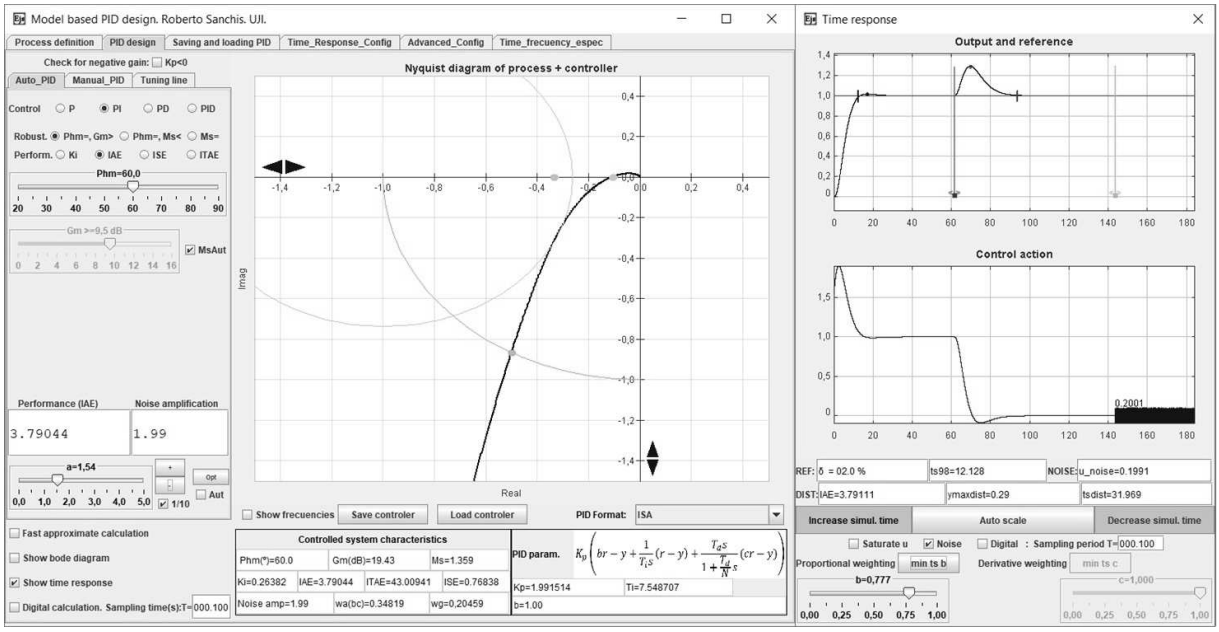

Figure 10. Example 1. PI controller.

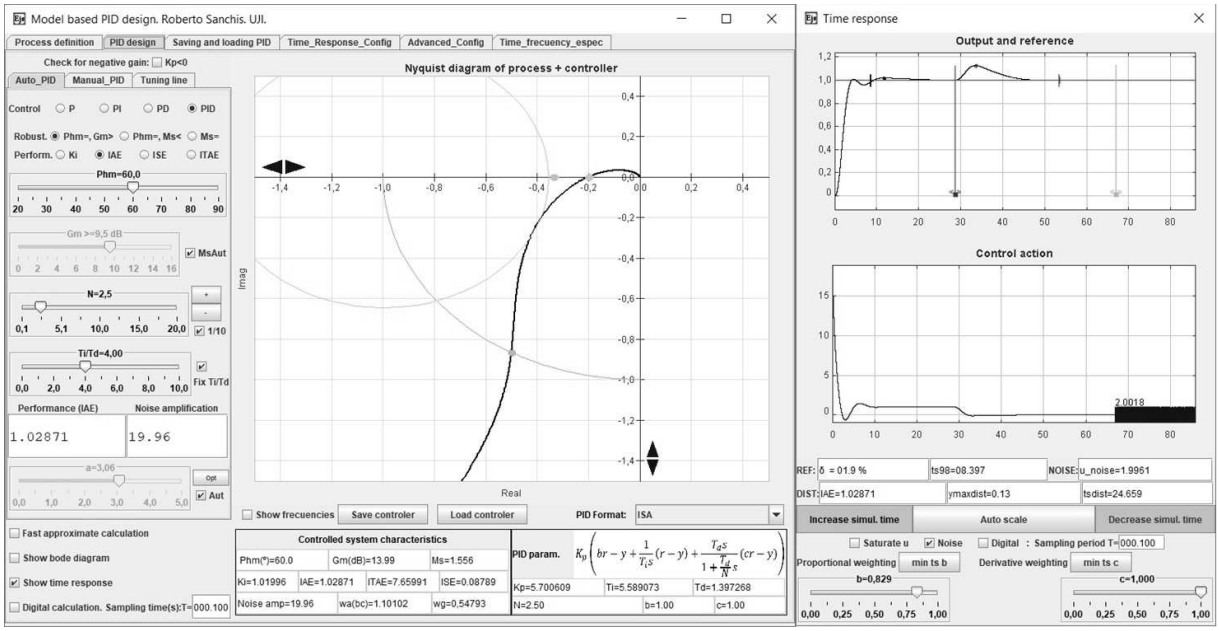

Figure 11. Example 1. PID controller with $T_i/T_d = 4$.

6.2 *Example2*

Consider a system modeled by the transfer function

$$G_2(s) = \frac{1}{(1+0.2s)^3} e^{-0.2s} \tag{14}$$

Assume that we are interested in reaching a velocity error below 0.5, that implies a value $K_i \geq 2$, with a robustness defined by $\Phi_m = 50$, $G_m \geq 7.7dB$, while we want to minimize the control effort due to noise. If we choose the performance index K_i and define the required robustness, using the tuning line tab and pressing the "M" button, the optimal PI is computed. A value

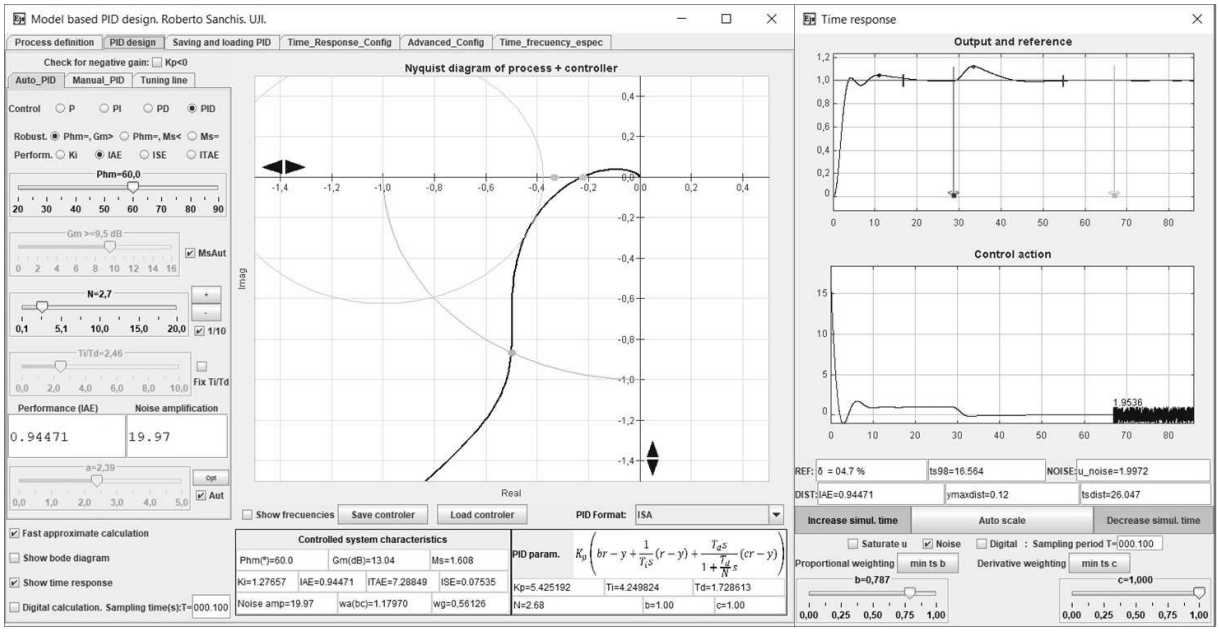

Figure 12. Example 1. PID controller with optimal T_i/T_d.

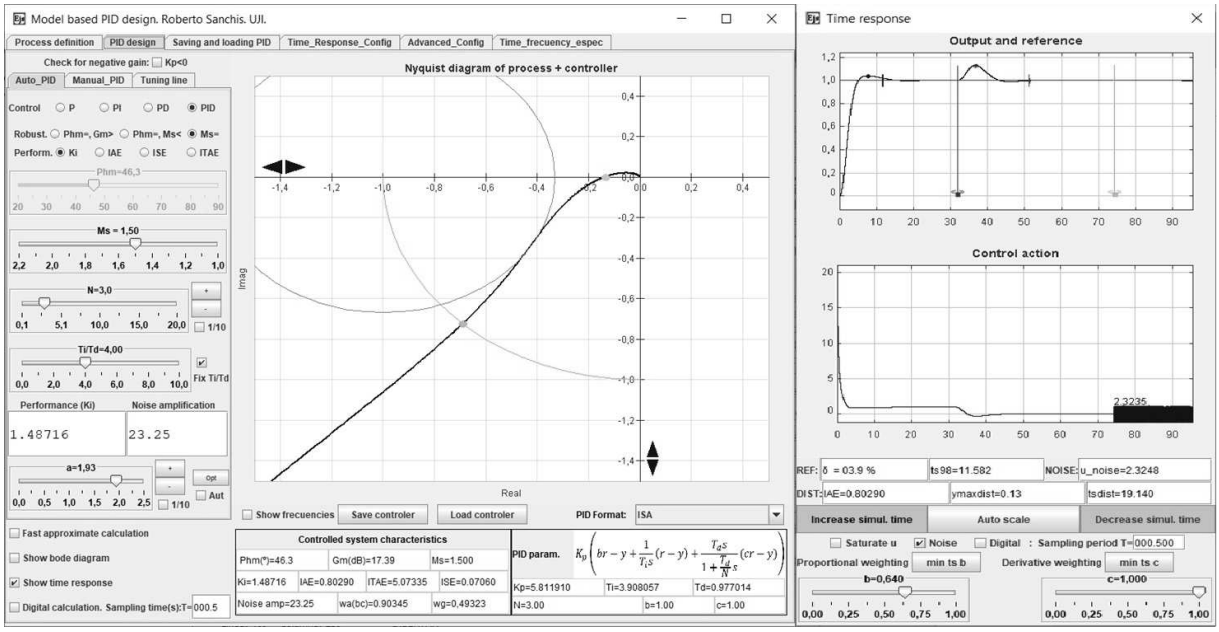

Figure 13. Example 1. PID controller for $M_s = 1.5$, with $T_i/T_d = 4$.

$K_i = 1.54 < 2$ is obtained , therefore we need a faster controller. Pressing button "R" we reach one point of the tuning line corresponding to a PID with $N = 3$, with a value $K_i = 2.11 > 2$. Moving a little bit to the left in the tuning line until a value of K_i slightly higher than 2 is obtained, we reach a PID controller with $N = 1.6$. The noise amplification is 2.17 (therefore, the control effort due to noise is 2.17 times the measurement noise). In this example, with a significant time delay, the gain margin constraint plays an important role, since the final gain margin is the one defined by the bound ($7.7dB$). Figure 15 shows the designed controller.

If we uncheck the box "Fix Ti/Td", and proceed in the same way, we find in the tuning line a PID with $N = 0.96$, an optimal value of $T_i/T_d = 1.77$, and a noise amplification of $1.39 << 2.17$. In this case, there is no cost in terms of robustness, because the phase and gain margins are

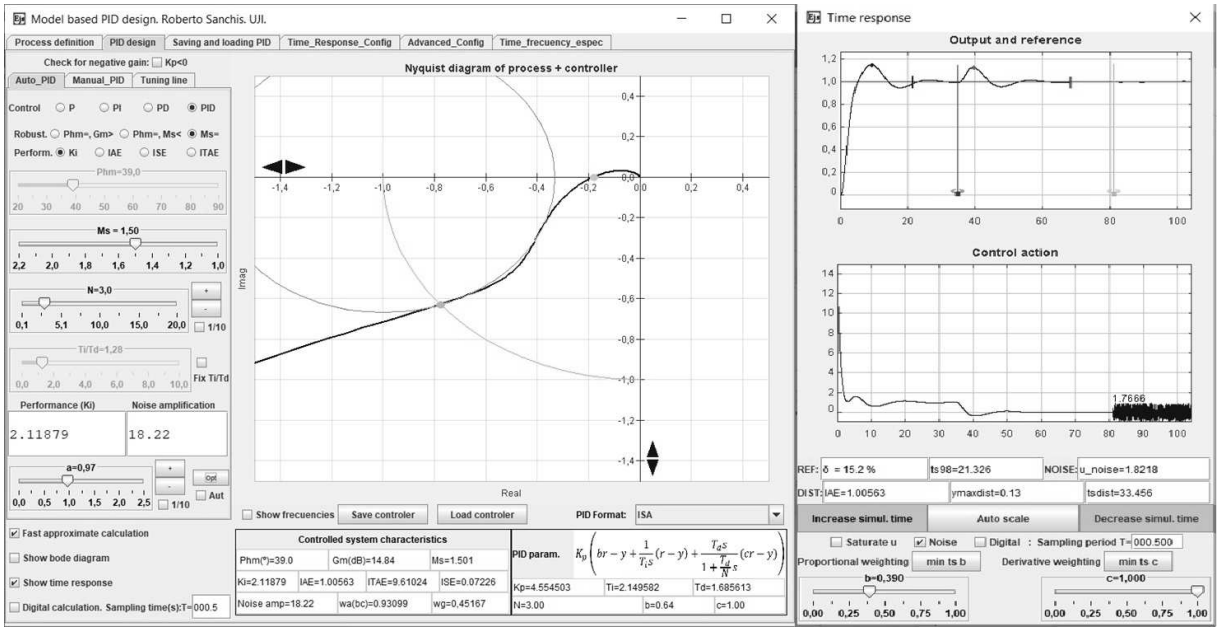

Figure 14. Example 1. PID controller for $M_s = 1.5$, with optimal T_i/T_d.

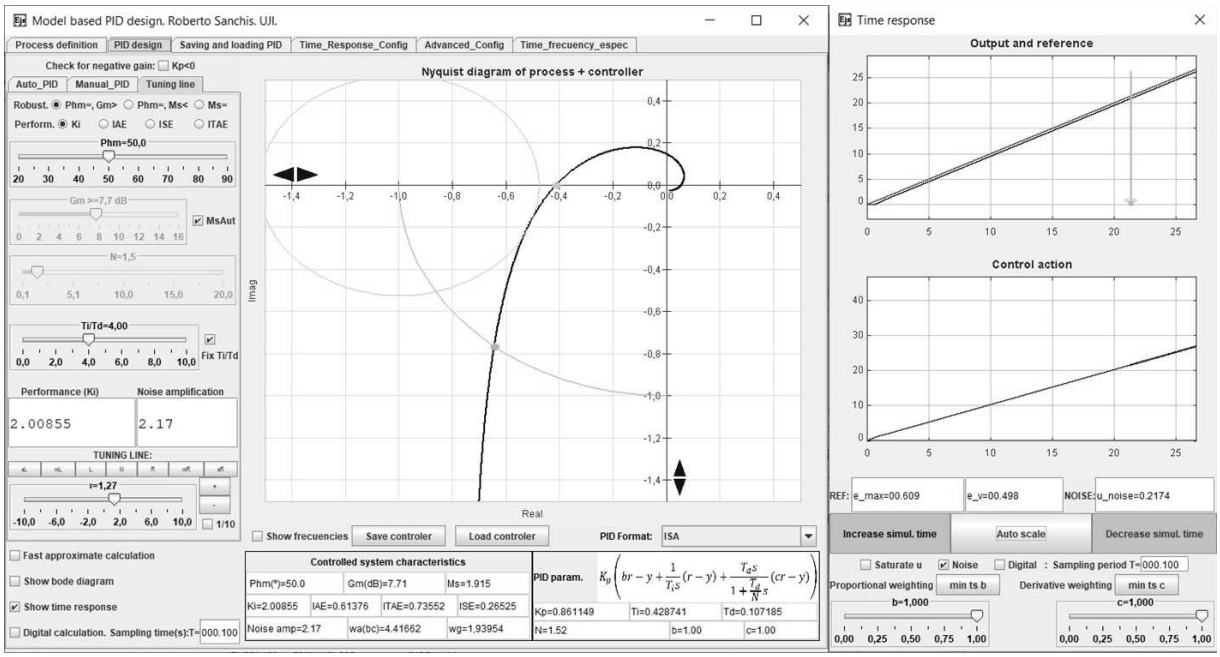

Figure 15. Example 2. PID controller with $T_i/T_d = 4$.

equal and M_s is lower, and hence, the robustness increases slightly. Figure 16 shows the designed controller.

6.3 *Example 3*

Consider a system modeled by the transfer function

$$G_3(s) = \frac{1 - 0.2s}{(1 + 0.1s)(1 + 0.2s)(1 + 0.3s)} \tag{15}$$

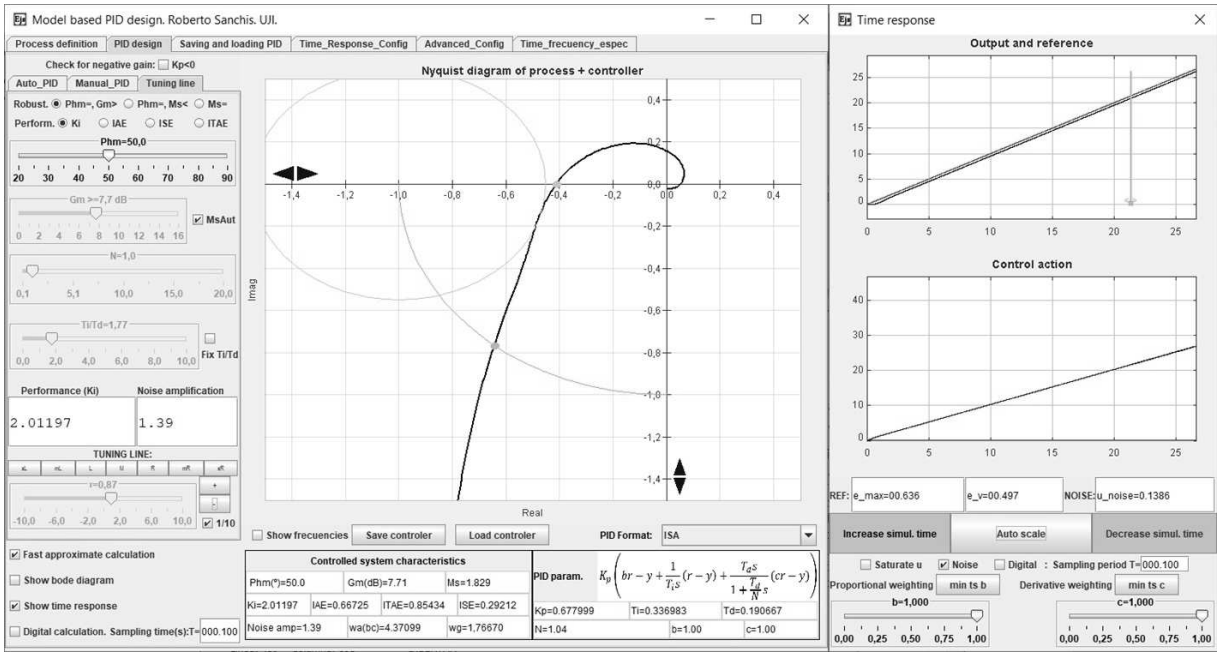

Figure 16. Example 2. PID controller with optimal T_i/T_d.

Assume we want to track a reference with a rising ramp, a constant interval and a descent ramp, as defined in figure 17. Assume the robustness constraint defined by $M_s = 1.5$, and that the control effort due to noise must be below 3 times the measurement noise. Assume also that the digital sampling time implementation of the controller is $T = 0.5s$.

If we design the controller in continuous time, using the tuning line, with $T_i/T_d = 4$, one obtains the PID shown in figure 18, with $N = 3.9$. However, if we check the box "Digital calculation", with a sampling time 0.5, we can see the real behavior of the digital implementation of the PID controller. Figure 19 shows that the real value of M_s is 2.66 instead of 1.5 as required, and the behavior is too oscillatory.

To solve this problem, we can calculate the controller in digital mode, to guarantee the required robustness using the sampled data model. If we design the controller with the check box "Digital calculation" on, and a sampling time $T = 0.5$ then the fastest point of the tuning line (corresponding to $N = 20$) has a noise amplification of 0.69<3, fulfilling the required robustness. Figure 20 shows the designed controller. In this example the optimization of T_i/T_d results in a slightly lower tracking error, with a slightly decrease in phase margin and gain margin, as shown in figure 21.

6.4 *Example 4*

Consider a system with an integrator, modeled by the transfer function

$$G_4(s) = \frac{1}{s(1+s)^2} \tag{16}$$

Assume that we are interested in minimizing the disturbance IAE, but we want a robustness defined by $\Phi_m = 60^o$, $G_m \geq 9.5dB$, while we want to maintain the high frequency noise amplification below 5. This means that the fluctuation of the control action due to noise will be five times the amplitude of measurement noise. Using the application. Using the tuning line tab, with a fixed value of $T_i/T_d = 4$, and pressing the "M" button, the optimal PI is computed. A noise amplification of $0.18 < 5$ is obtained , therefore we need a faster controller. Pressing button

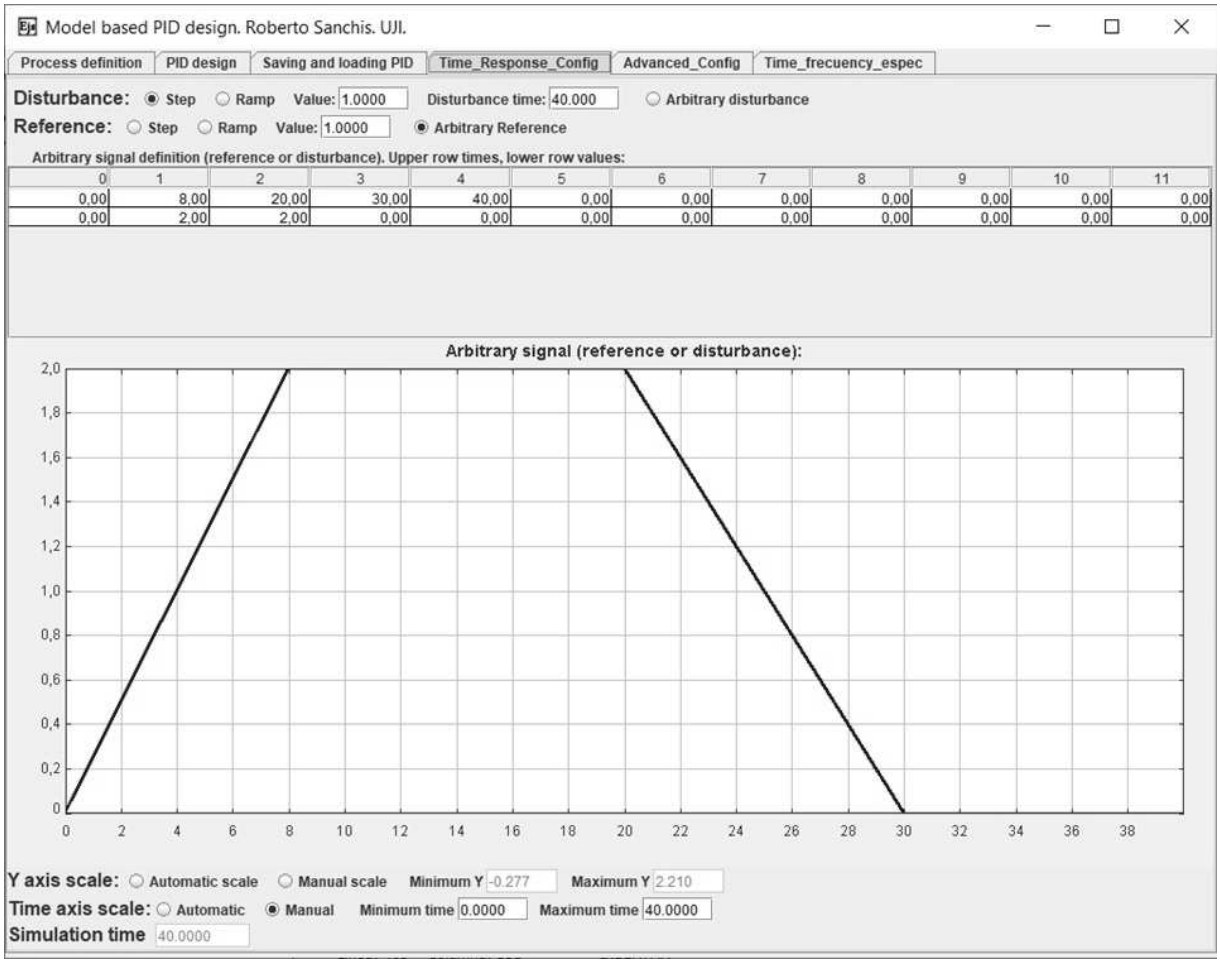

Figure 17. Example 3. Reference signal.

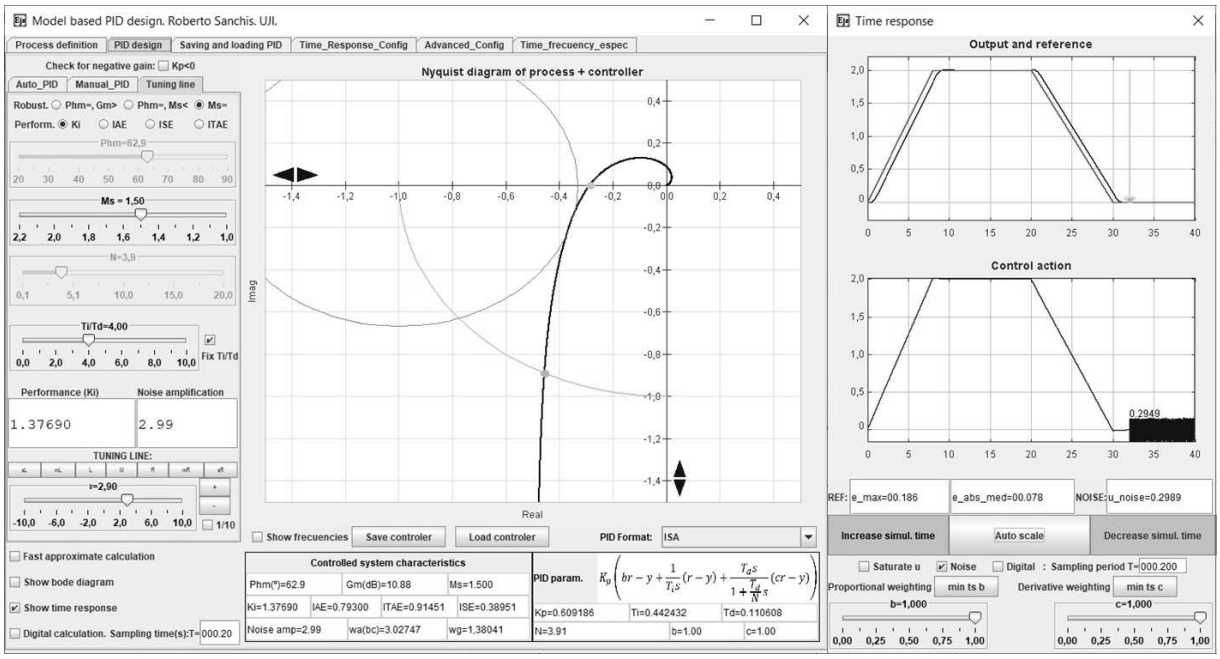

Figure 18. Example 3. CT PID controller with $T_i/T_d = 4$.

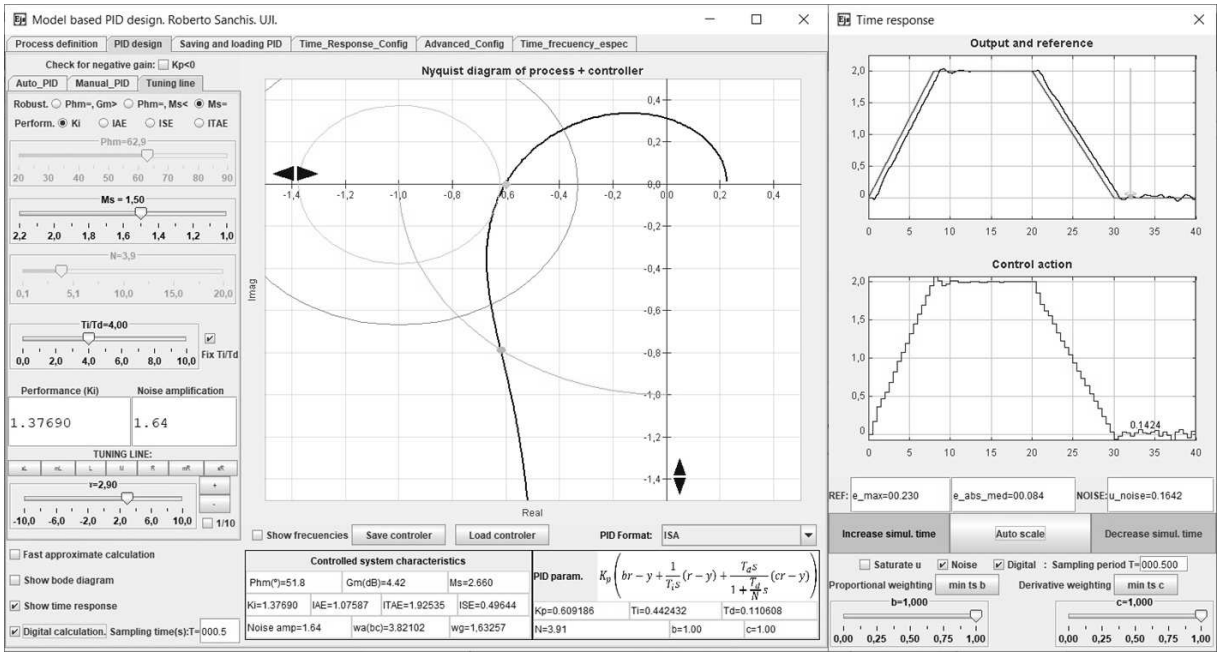

Figure 19. Example 3. Digital implementation of CT PID controller with $T_i/T_d = 4$.

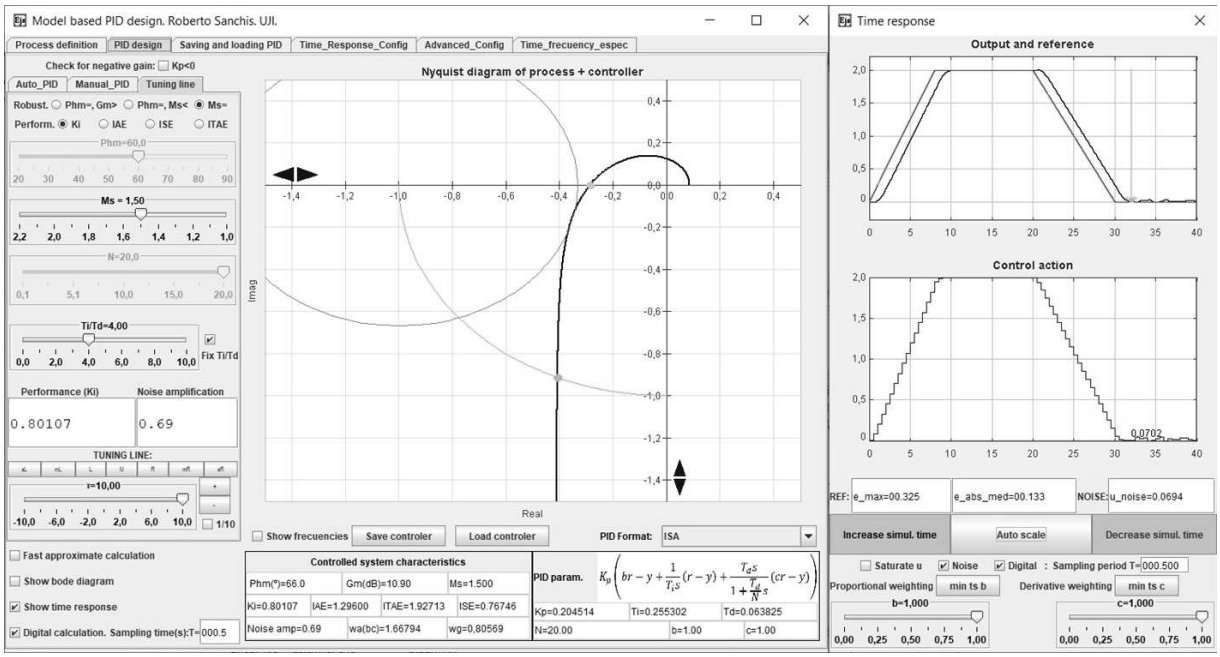

Figure 20. Example 3. Digitally designed PID controller with $T_i/T_d = 4$.

"mR" we reach one point of the tuning line corresponding to a PID with $N = 8.7$, with a noise amplification $6.56 > 5$. Moving a little bit to the left in the tuning line until a value of noise amplification slightly lower than 5 is obtained, we reach a PID controller with $N = 6.9$. The resulting IAE is 11.5. Figure 22 shows the designed controller characteristics. We can see that the gain margin constraint do not affect the design, because it is fulfilled in excess. In the time response window, we have tuned by hand the weighting factors b and c to reduce the overshoot in the reference response. If we solve the same problem unchecking the box "Fix Ti/Td", the obtained PID has $T_i/T_d = 1.62$, and an IAE=9.5, as shown in Figure 23. The IAE is smaller,

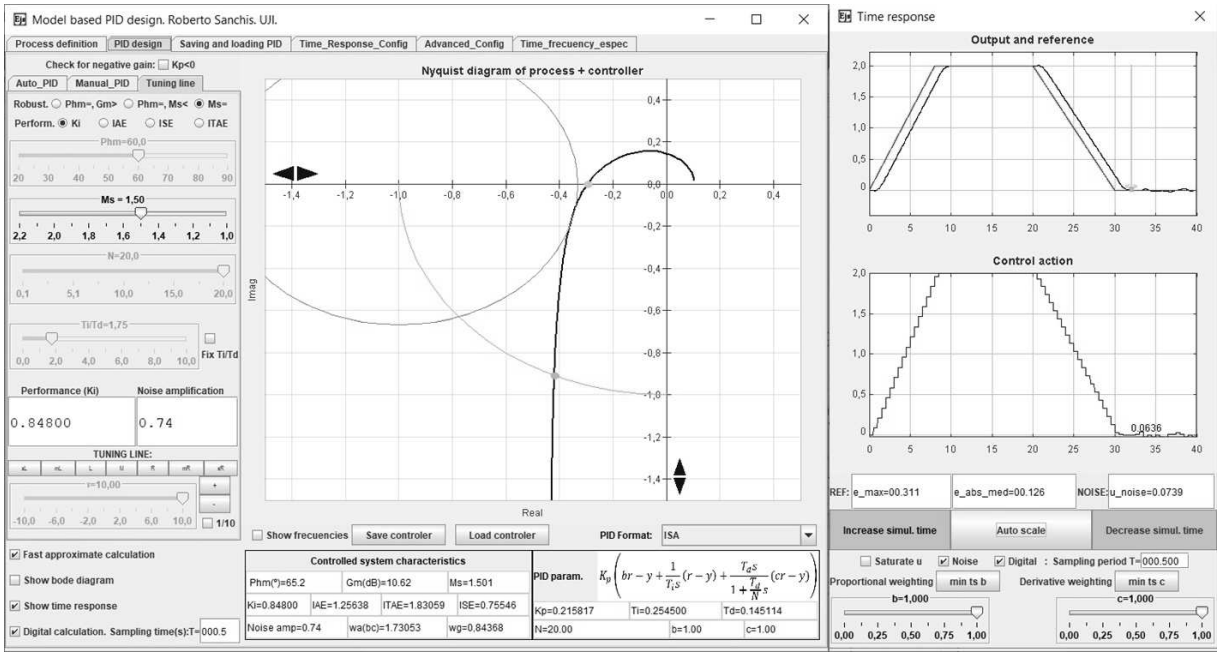

Figure 21. Example 3. Digitally designed PID controller with optimal T_i/T_d.

but at the expense of a less robust response (higher M_s), and a more oscillatory response.

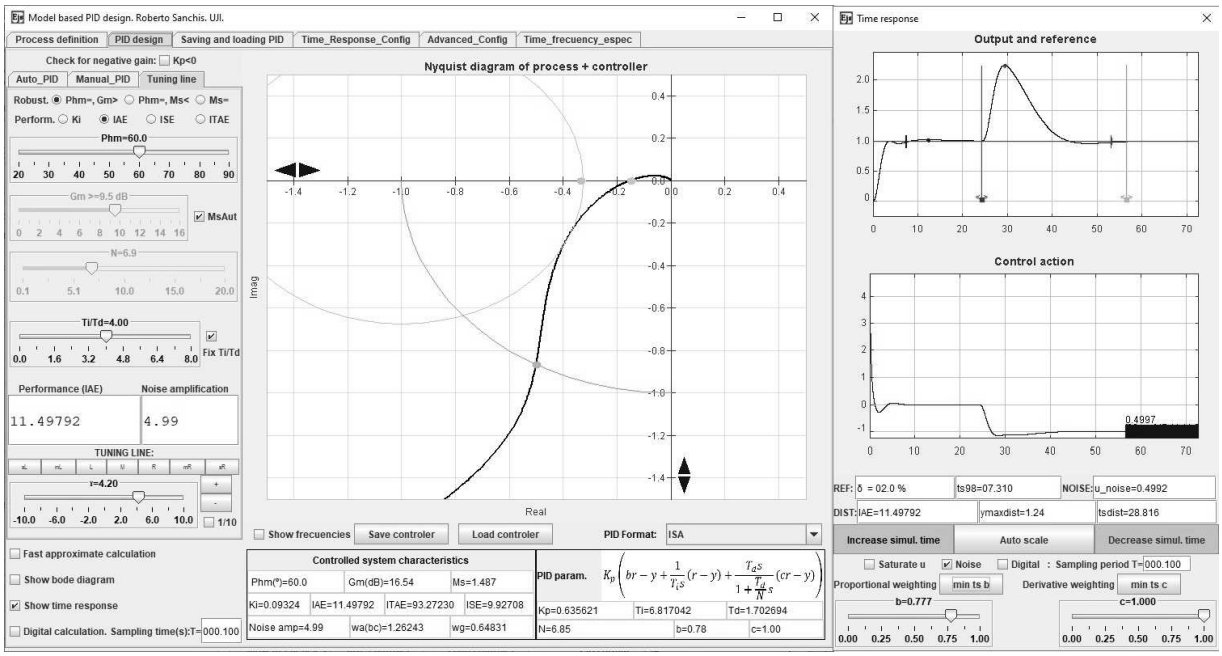

Figure 22. Example 4. PID controller with $T_i/T_d = 4$.

6.5 *Example 5*

Consider a system with a very dominant time delay, modeled by the transfer function

$$G_5(s) = \frac{1}{(1+s)^2} e^{-5s} \tag{17}$$

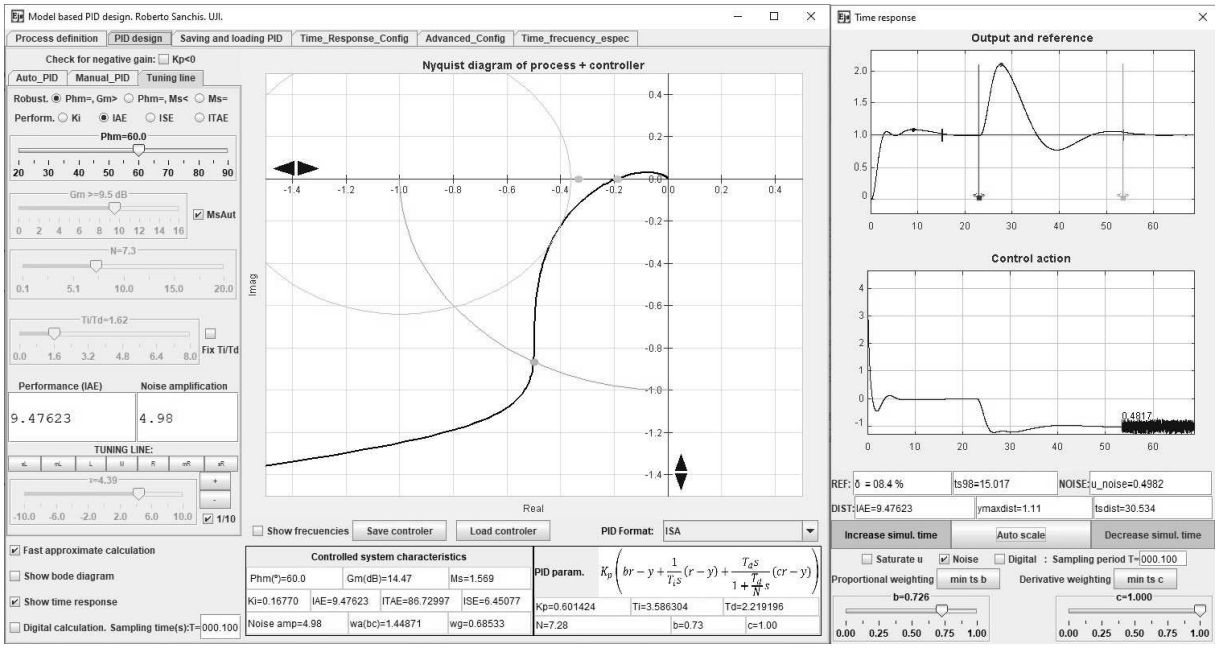

Figure 23. Example 4. PID controller with arbitrary T_i/T_d.

Assume that we are interested in minimizing the disturbance IAE, but we want a robustness defined by $\Phi_m = 60^o$, $G_m \geq 9.5dB$. Using the tuning line tab, with a fixed value of $T_i/T_d = 4$, and pressing the "M" button, the optimal PI is computed. The resulting performance and noise amplification are $IAE = 10.63$ and $C(\infty) = 0.1$. If we select a point on the right of the tuning line (for example point R, with the box "Fix Ti/Td" checked), the resultant PID has $IAE = 9.74$ and $C(\infty) = 0.64$. If compared to a lag dominant process (as in example 1), the improvement in performance thanks to the derivative term is very small compared to the increase in noise amplification. This is a well known fact about delay dominant processes. Taking the most right hand side point of the tuning line (point xR), one obtains $IAE = 9.74$ and $C(\infty) = 3.37$, i.e. no improvement in performance but a high increase in noise amplification. If we let the ratio T_i/T_d to vary freely, for the point R in the tuning line we obtain $IAE = 7.75$ and $C(\infty) = 1.2$, that is a slightly higher improvement of performance with respect the PI (point M), but still much less than the improvement observed in lag dominant processes (example 1).

7 Conclusions

In this paper we have presented a procedure for tuning PID controllers to find the optimal compromise between robustness, performance and control effort due to high frequency noise. The procedure is based on the fixed robustness tuning line concept, in which the derivative filter parameter is a key point in finding the compromise between control effort due to noise and performance. For a given robustness constraint, each point of the tuning line represents a PID controller that optimizes the compromise between performance and control effort due to noise. With the approach is easy to find the controller that optimizes the performance while fulfilling a prescribed robustness constraint and a maximum permitted control effort due to noise.

We have also presented a free software tool that implements the proposed tuning procedure. The tool has been developed in Java, with Easy Java Simulations, and can be freely downloaded from https://sites.google.com/a/uji.es/freepidtools/, where other free tools can also be found (for system identification for example). The application accepts three different forms to define the robustness, and four different performance indexes for the optimization. The tuning line for fixed robustness has been implemented in the application. The tool lets the user to easily

choose one point of the tuning line, reaching the desired compromise between speed response and control effort due to noise. This allows to solve immediately tuning problems of the type: find the PID controller that minimizes the performance (for example the IAE), fulfilling a robustness constraint, and a required control effort due to noise. Furthermore, the tool allows to calculate the controller in continuous time (both system and controller are continuous time transfer functions), or in digital mode, using the ZOH discrete system model and a discretized PID, for the provided sampling period. This is especially useful when the implementation period can not be sufficiently small, compared to the system dynamics.

Acknowledgements

This work has been supported by MICINN project number TEC2015-69155-R from the Spanish government.

References

Amirinejad, M., Eslami, M., and Noori, A.A. (2014), "Automatic PID Controller Parameter Tuning Using Bees Algorithm," *International journal of scientific and engineering research*, 5.

Astrom, K.J., Panagopoulos, H., and Hagglund, T. (1998), "Design of PI Controllers based on Non-Convex Optimization," *Automatica*, 34, 585–601.

Boyd, S., Hast, M., and Aström, K.J. (2016), "MIMO PID tuning via iterated LMI restriction," *International Journal of Robust and Nonlinear Control*, 26, 1718–1731.

D'azzo, J., and Houpis, C., *Feedback Control Systems*, Mc Graw Hill (1966).

Díaz, J.M., Costa-Castelló, R., Muñoz, R., and Dormido, S. (2017), "An Interactive and Comprehensive Software Tool to Promote Active Learning in the Loop Shaping Control System Design," *IEEE Access*, 5, 10533–10546.

Dorf, R.C., and Bishop, R., *Modern Control Systems*, Prentice Hall (2007).

Franklin, G., Powell, J., and Emami-Naeini, A., *Feedback Control of Dynamic Systems*, Prentice Hall (2005).

Garrido, J., Ruz, M.L., Morilla, F., and Vázquez, F. (2018), "Interactive Tool for Frequency Domain Tuning of PID Controllers," *Processes*, 6, 197.

Guzmán, J., Aström, K., Dormido, S., Hägglund, T., Berenguel, M., and Piguet, Y. (2008), "Interactive Tool for Frequency Domain Tuning of PID Controllers," *IEEE Control Systems Magazine*, pp. 118–134.

H. Panagopoulos, K.A., and T.Hagglund, (2002), "Design of PID controllers based on constrained optimization," *IEE Proc.-Control Theory Appl.*, 149, 32–40.

Hast, M., Aström, K.J., Bernhardsson, B., and Boyd, S. (2013), "PID design by convex-concave optimization," in *2013 European Control Conference (ECC)*, IEEE, pp. 4460–4465.

Ho, W.K., Lim, K.W., and Wen, X. (1998), "Optimal Gain and Phase Margin Tuning for PID Controllers," *Automatica,*, 34, 1009–1014.

Hwang, C., and Hsiao, C.Y. (2002), "Solution of a non-convex optimization arising in PI/PID control design," *Automatica,*, 38, 1895–1904.

Levy, D., Lu, Y., Yan, D., Zhou, M., and Chen, S. (2019), "Particle Swarm Optimization of Real-Time PID Controllers," *Handbook of Real-Time Computing*, Springer.

Liu, G.P., and Daley, S. (1999), "Optimal-tuning PID controller design in the frequency domain with application to a rotary hydraulic system," *Control Engineering Practice,*, 7, 821–830.

Madhuranthakam, C.R., Elkamel, A., and Budman, H. (2008), "Optimal tuning of PID controllers for FOPTD, SOPTD and SOPTD with lead processes," *Chemical Engineering and Processing: Process Intensification,*, 47, 251–264.

Mercader, P., Aström, K.J., Baños, A., and Hägglund, T. (2016), "Robust PID design based

on QFT and convex–concave optimization," *IEEE transactions on control systems technology*, 25, 441–452.

Ogata, K., *Modern Control Engineering*, Prentice Hall (2003).

Phillips, C., and Harbor, R., *Feedback Control Systems*, Prentice Hall (1999).

Reynoso-Meza, G., Sanchis, J., Blasco, X., and Martinez, M. (2013), "Algoritmos evolutivos y su emnpleo en el ajuste de controladores del tipo PID: estado Actual y Perspectivas," *Revista Iberoamericana de Automatica e Informatica Industrial*, pp. 251–268.

Romasevych, Y., Loveikin, V., and Makarets, V. (2020), "Optimal Constrained Tuning of PI-Controllers via a New PSO-Based Technique," *International Journal of Swarm Intelligence Research*, 11, 87–105.

Sanchis, R., Romero, J., and Balaguer, P. (2010), "Tuning of PID controllers based on simplified single parameter optimisation," *International Journal of Control*, 83, 1785–1798.

Tavakoli, S., Griffin, I., and Fleming, P. (2005), "Robust PI Controller for Load Disturbance Rejection and Setpoint Regulation," *IEEE International Conference on Control Applications*, pp. 1015–1020.

Toscano, R. (2005), "A simple robust PI/PID controller design via numerical optimization approach," *Journal of Process Control,*, 15, 81–88.

Zhao, S.Z., Iruthayarajan, M., Baskar, S., and Suganthanr, P. (2011), "Multi-objective robust PID controller tuning using two lbests multi-objective particle swarm optimization," *Information Sciences*, 181, 3323–3335.